

Multi-Criteria Evaluation of Partitioning Schemes for Real-Time Systems

Irina Lupu
irina.lupu@ulb.ac.be

Pierre Courbin
courbin@ece.fr

Laurent George
lgeorge@ece.fr

Joël Goossens
joel.goossens@ulb.ac.be

April 22, 2010

Abstract

In this paper we study the partitioning approach for multiprocessor real-time scheduling. This approach seems to be the easiest since, once the partitioning of the task set has been done, the problem reduces to well understood uniprocessor issues. Meanwhile, there is no optimal and polynomial solution to partition tasks on processors. In this paper we analyze partitioning algorithms from several points of view such that for a given task set and specific constraints (processor number, task set type, etc.) we should be able to identify the best heuristic and the best schedulability test. We also analyze the influence of the heuristics on the performance of the uniprocessor tests and the impact of a specific task order on the schedulability. A study on performance difference between Fixed Priority schedulers and EDF in the case of partitioning scheduling is also considered.

1 Introduction

Uniprocessor scheduling has been widely studied over the last few decades. However, a single processor can no longer satisfy today's computational demands, as the miniaturization of integrated circuits reaches its physical limits [18]. Thus, a valid solution to supply sufficient resources is the use of multiprocessor platforms.

There are two main techniques for multiprocessor scheduling. Given a task set τ and a m -processors platform π , one can choose between:

- partitioning scheduling: τ is divided into a number of disjoint subsets less than or equal to the number of processors of the platform. Each of these subsets is assigned to one processor. Uniprocessor scheduling policies are then used locally on each processor.
- global scheduling: at each instant t , the m highest priority tasks are executed on the platform allowing the migration of tasks from one processor to another with the restriction that a task cannot be executed on different processors at the same time.

The two techniques are incomparable: there are task sets which are schedulable by partitioning but not by global approach and conversely [7]. However, today's state of the art declares that the performance of multiprocessor partitioning scheduling has not yet been overtaken by the global approach despite its considerable improvements over the last few years [5, 4]. Even if, in certain scenarios, the worst case achievable utilization is the same for both techniques when using classical schedulers like EDF or Fixed Priority (FP) [1], in the average case, partitioning schemes seem to behave generally better than global ones. In this paper, we focus on the partitioning approach.

Partitioning a task set is equivalent to the Bin-Packing problem: how to place n objects of different sizes in m boxes. This problem is known to be NP-hard. The only known solution for this kind of problem is to enumerate all possible configurations and verify their correctness one by one. Suboptimal possible solutions have been proposed in the literature and are known as *partitioning heuristics*.

Previously proposed partitioning algorithms (with both FP or EDF) [6] are composed of a heuristic, a uniprocessor schedulability test and, very often, a task sorting criterion. Though, there does not exist an exhaustive comparative evaluation of all of them. The present state of the art compares *some* partitioning algorithms based on the First Fit heuristic (a task is assigned to the first processor that can feasibly schedule it) and the decreasing utilization sorting criterion [5, 4]. The evaluation is usually done in terms of success ratio (the ratio of successfully scheduled task sets over all task sets considered).

Even though the success ratio is a very important evaluation criterion, other measures like the number of processors used and the available unutilized processor capacity are important since the design of real-time systems is often constrained by: the number of available processors, whether there is a risk to encounter software or hardware errors during execution, etc. [15]. *In this research we establish some directions for deciding which heuristic, which schedulability test and which task set sorting criterion to choose according to such design constraints.*

This research. Our goal is to identify the best partitioning algorithm for different scenarios; for this purpose, we combine 4 partitioning heuristics (First Fit, Best Fit, Next Fit, Worst Fit), 8 task sorting criteria (decreasing or increasing period, utilization, etc.) and 8 schedulability tests associated to FP or EDF scheduling policies. Our aim is to obtain a nearly exhaustive list of possible partitioning algorithms. Their performance will be compared with the Optimal Partitioning Algorithm (OPA in the following) according to each one of the 3 criteria: success ratio, number of processors used and available unutilized processor capacity.

Since we implemented the OPA, we also use simulation to evaluate the suboptimality of heuristics (the performance gap according to the optimal task assignment, in terms of success ratio) and the influence of this suboptimality on the performance of the tests. In this paper we also study the influence of the task sorting criteria on the schedulability of each heuristic and each test.

Paper Organization. The remainder of this paper is organized as follows: in Section 2, we describe the system model and other concepts used in this paper. Section 3 presents the parameters of the study (partitioning heuristics, task sorting criteria and schedulability tests which build the partitioned scheduling algorithms). Section 4 describes the task generation methodology and Section 5 gives the results of the simulations. In Section 6, we present the conclusions and the future work in Section 7.

2 System Model

In this paper we refer to the sporadic task model. A sporadic task τ_i is defined by (C_i, T_i, D_i) where C_i is the worst-case execution time (WCET), T_i is the minimum inter-arrival time (also called the period) and D_i is the relative deadline (a task released at time t must be executed by its absolute deadline $t + D_i$). A task can be instantiated an infinite number of times. An instance of a task is called a *job*.

The set $\tau = \{\tau_1, \dots, \tau_n\}$ is composed of n sporadic tasks.

For FP schedulers we consider tasks in decreasing order of their priority: τ_1 is the highest priority task and τ_i has higher priority than τ_{i+1} .

We can distinguish between three kinds of sporadic task sets: implicit deadlines ($D_i = T_i, \forall i = 1, \dots, n$), constrained deadlines ($D_i \leq T_i, \forall i = 1, \dots, n$) or arbitrary deadlines (no restriction on deadlines).

Each task is characterized by a utilization factor $u_i \stackrel{\text{def}}{=} \frac{C_i}{T_i}$ and a density $\lambda_i \stackrel{\text{def}}{=} \frac{C_i}{\min\{D_i, T_i\}}$.

Each task set is characterized by the following measures:

- task set utilization $U_\tau \stackrel{\text{def}}{=} \sum_{i=1}^n u_i$ and task set density $\Lambda_\tau \stackrel{\text{def}}{=} \sum_{i=1}^n \lambda_i$;
- for a given t , the *Request Bound Function* (RBF) represents the upper bound of the work load generated by all tasks with activation times included in the interval $[0, t)$: $\text{RBF}(\tau, t) \stackrel{\text{def}}{=} \sum_{i=1}^n \lceil \frac{t}{T_i} \rceil \cdot C_i$. An approximation for RBF [14] is: $\text{RBF}^*(\tau, t) \stackrel{\text{def}}{=} \sum_{i=1}^n (C_i + u_i \cdot t)$.
- for a given t , the *Demand Bound Function* (DBF) represents the upper bound of the work load generated by all tasks with activation times and absolute deadlines in the same interval $[0, t]$: $\text{DBF}(\tau, t) \stackrel{\text{def}}{=} \sum_{i=1}^n \max\{0, 1 + \lfloor \frac{t-D_i}{T_i} \rfloor\} \cdot C_i$. An approximation for DBF [8] is: $\text{DBF}^*(\tau, t) \stackrel{\text{def}}{=} \sum_{i=1}^n \mathcal{A}_i(t)$ with $\mathcal{A}_i(t) \stackrel{\text{def}}{=} (C_i + (t - D_i) \cdot u_i)$ if $t \geq D_i$ and 0, if not.

In the following, we consider π a platform with m identical processors: $\pi = \{\pi_1, \dots, \pi_m\}$. By $\tau(\pi_j)$ we will denote the task subset assigned to the processor π_j .

3 Parameters of the Study

Our study includes *all possible combinations* between a series of 4 partitioning *heuristics*, 8 *task sorting criteria* and 8 *schedulability tests* associated with FP or EDF scheduling policies. For each combination, we evaluate the *performance* according to three criteria. In this section, we introduce each of these parameters.

3.1 Heuristics

As mentioned previously, the Bin-Packing problem (which lies at the basis of the partitioning technique) is known to be NP-Hard. Some heuristics have been proposed in the literature in order to solve it and they are in the focus of this paper. All of them imply a sequential assignment of tasks to processors:

- First Fit (FF): a task is assigned to the first processor, starting from π_1 , which verifies the schedulability test after the assignment.
- Best Fit (BF): a task is assigned to the processor which verifies the schedulability test after assignment and which minimizes the remaining processor capacity.
- Worst Fit (WF): a task is assigned to the processor which verifies the schedulability test after assignment and which maximizes the remaining processor capacity.
- Next Fit (NF): a task is assigned to the first processor in the range $\{\pi_j, \dots, \pi_m\}$ that verifies the schedulability test after the assignment (π_j is the current processor). The procedure starts from π_1 .

Impact of the Heuristic. Heuristics can have a great impact on the solution. Consequently, it is important to know in which of the following cases the problem falls:

1. we want to minimize the number of processors used;
2. we want to provide enough time slack (i.e., leave spare capacity on processors) in order to handle system overload.

3.2 Task Sorting Criteria

Often, the partitioning algorithms proposed in the literature include task sorting criteria with the purpose of increasing their success ratio. In this paper we analyze the general influence of 8 such criteria *increasing/decreasing*: deadline, density, period and utilization.

3.3 Schedulability Tests

The schedulability tests considered in this paper are necessary and sufficient (i.e., exact) (NST) or sufficient (ST) with polynomial (P), pseudo-polynomial (PP). The tests are designed for implicit deadlines (ID), constrained deadlines (CD) or arbitrary deadlines (AD). In our study, the arbitrary deadline tests are applied to the constrained case. In the following, the name of each test represents the associated priority assignment rule followed by the initials of its authors.

3.3.1 EDF-based schedulability tests

For EDF two *exact* and a third one only *sufficient* schedulability tests are taken into account:

EDF-LL [17] (NST-P-ID): $U_\tau \leq 1$;

EDF-BHR [9] (NST-PP-AD): $\text{Load}(\tau) \stackrel{\text{def}}{=} \sup_{t \geq 0} \frac{\text{DBF}(\tau, t)}{t} < 1$. In order to reduce the number of instants t to consider, we use the following expression [15]: $\text{Load}(\tau) = \max\{U_\tau, \sup_{t \in [D_{\min}, P)} \frac{\text{DBF}(\tau, t)}{t}\}$, where $D_{\min} \stackrel{\text{def}}{=} \min\{D_1, \dots, D_n\}$ and $P \stackrel{\text{def}}{=} \text{lcm}\{T_1, \dots, T_n\}$ [15]. As the DBF changes its value only at instants corresponding to absolute deadlines, $\text{Load}(\tau)$ will be computed for each $t \in \bigcup_{j=1}^n \{D_j + k_j \cdot T_j, 0 \leq k_j \leq \lceil \frac{P-D_j}{T_j} \rceil - 1\}$.

EDF-BF [8] (ST-P-AD): $\forall \tau_i \in \tau, D_i - \text{DBF}^*(\tau \setminus \{\tau_i\}, D_i) \geq C_i$ and $1 - \sum_{\tau_j \in \tau, \tau_j \neq \tau_i} u_j \geq u_i$.

As previously mentioned, the EDF-BHR test is exact but it has a pseudo-polynomial complexity [9]. A solution to this problem has been proposed [15]: reduce the number of analyzed instants (in the interval $[D_{\min}, P)$) by formulating the problem as a Linear Programming one; in order to solve it use the *simplex* algorithm. Even though the number of instants is significantly reduced, the EDF-BF test offers a simpler (polynomial) solution to verify the schedulability of the task sets, but it is only a sufficient test.

3.3.2 FP-Based Schedulability Tests

FP schedulers assign fixed priorities to tasks before the execution of the system and the jobs inherit the priority of the task that generated them. During our simulations we consider DM (for AD et CD tests) and RM (for ID tests) schedulers [17, 3].

DM-ABRTW [2] (NST-PP-CD): DM-test based on the response-time analysis: $\forall \tau_i \in \tau, r_i \leq D_i$, where r_i is τ_i 's *worst case response time*. For constrained deadlines, r_i is determined for the first activation of the task in the synchronous scenario as a solution of the equation: $r_i = C_i + \sum_{j=1}^{i-1} \lceil \frac{r_i}{T_j} \rceil \cdot C_j$ computed as follows: $W_0 = C_i$ and $W_{k+1} = C_i + \sum_{j=1}^{i-1} \lceil \frac{W_k}{T_j} \rceil \cdot C_j$ until $W_{k+1} = W_k$ or $W_k > D_i$.

DM-FBB [13] (ST-P-AD): $\forall \tau_i \in \tau, D_i - \text{RBF}^*(\tau - \tau_i, D_i) \geq C_i$ and $1 - \sum_{\tau_j \in \tau, \tau_j \neq \tau_i} u_j \geq u_i$.

RM-LL [17, 12] (ST-P-ID): $U_\tau \leq n(\sqrt[n]{2} - 1)$.

RM-BBB [10] (ST-P-ID): $\prod_{i=1}^n (u_i + 1) \leq 2$.

RM-LMM [16] (ST-P-ID): If $1 \leq r_\pi \stackrel{\text{def}}{=} \frac{T_{\max}(\tau)}{T_{\min}(\tau)} < 2$

(where $T_{\max}(\tau) \stackrel{\text{def}}{=} \max\{T_1, \dots, T_n\}$ and $T_{\min}(\tau) \stackrel{\text{def}}{=} \min\{T_1, \dots, T_n\}$), then τ is RM schedulable on processor π if $U_\tau \leq B(r_\pi, n)$ with $B(r_\pi, n) \stackrel{\text{def}}{=} n \cdot (\sqrt[n]{r_\pi} - 1) + \frac{2}{r_\pi} - 1$. For the case $r_\pi \notin [1, 2)$ a scaling algorithm is presented in [16] such that, if τ' is the task set obtained after scaling: τ is schedulable if and only if τ' is schedulable.

DM-ABRTW is an exact test, but it has a pseudo-polynomial complexity. DM-FBB is a uniprocessor schedulability test which has been proposed especially for partitioned scheduling [13]. Simulations show that when DM-ABRTW and DM-FBB are associated with the FF heuristic, their behavior is incomparable: for constrained deadline task sets DM-ABRTW behaves better, but for the case of arbitrary deadlines, DM-FBB has a higher success rate than DM-ABRTW.

RM-LL is the popular Liu & Layland FP test. For a sufficiently large number of tasks in the set, this test has an utilization bound of $\ln 2$ ($\cong 0.69$) [17]. However, RM can generally schedule task sets with a utilization of 88% (this shows the pessimism of the RM-LL test). RM-BBB test is also polynomial, but it establishes a higher task set utilization bound than RM-LL. Simulations show [10] that RM-BBB schedules $\sqrt{2}$ more task sets than RM-LL as the number of tasks grows. The last FP schedulability test, RM-LMM has already been applied in multiprocessor scheduling and compared with RM-LL when as-

sociated with the FF heuristic [16]. Simulations show that RM-LMM's average processor utilization is comparable to that of an exact test: 96%. In the same scenario, the RM-LL average processor utilization is 75%.

For BF and WF partitioning heuristics, the schedulability tests determine the expression used to compute the unutilized processor capacity when tasks are assigned to processors: $1 - \Lambda_\tau$ for EDF-LL, EDF-BF, DM-ABRTW, DM-FBB, RM-LL, RM-BBB and RM-LMM; and $1 - \text{Load}(\tau)$ for EDF-BHR, since this is the only test based on $\text{Load}(\tau)$.

3.4 Performance Criteria

Each combination of the previously mentioned parameters is evaluated according to several performance criteria:

- *Success Ratio*: with this criterion we determine which combination schedules the largest number of task sets. It is defined as follows:

$$\frac{\text{number of task sets successfully scheduled}}{\text{total number of task sets}}$$

- *Number of processors used* defined as number of processors where at least one task is assigned if m is sufficient.
- *Average value of processor spare capacity*: the spare capacity on processors is computed with the expression $1 - \text{Load}(\tau(\pi_j))$ for EDF-BHR and $1 - \Lambda_{\tau(\pi_j)}$ for the other schedulability tests.

3.5 Optimal Partitioning Algorithm

The OPA is designed as follows: enumerate all possible task assignments on processors and test them one by one. In terms of schedulability tests, for EDF, we associate this optimal assignment with the exact test of EDF-BHR (OPA[EDF]). For FP, we use the exact test of DM-ABRTW (OPA[FP]).

4 Task Generation Methodology

The task generation methodology used in this paper is based on the one presented in [5]. However, in our case, task generation is adapted to each type of deadline considered. In the following, $k_i \in \{D_i, T_i\}$ and $\rho_i \in \{u_i, \lambda_i\}$:

- k_i is uniformly chosen from $[1, 100]$;
- ρ_i (truncated between 0.001 and 0.999) has the following distributions:
 1. Uniform distribution between $\frac{1}{k_i}$ and 1;
 2. Bimodal distribution: heavy tasks have a uniform distribution between 0.5 and 1, light tasks have a uniform distribution between $\frac{1}{k_i}$ and 0.5; the probability of a task being heavy is $\frac{1}{3}$;
 3. Exponential distribution of mean 0.25;
 4. Exponential distribution of mean 0.50.

For implicit deadline task sets, $(k_i, \rho_i) = (T_i, u_i)$ and for constrained deadlines $(k_i, \rho_i) = (D_i, \lambda_i)$. In this last case, the period T_i is uniformly chosen from $[D_i, 100]$.

We consider 4-processor identical platforms.

Task systems are generated so that those obviously unfeasible ($U_\tau > m$) or trivially schedulable ($m = n$ and $\forall i \in [1, n], u_i \leq 1$, meaning the capacity of one processor of the identical platform) are not considered during simulations:

Step 1: Initially, we generate a system which contains $m + 1$ tasks and test it.

Step 2: We add tasks to the system and repeat the tests until the density of the system exceeds m (the capacity of the identical platform).

For our simulations, we generated 10^6 task sets uniformly chosen from the distributions mentioned above with implicit and constrained deadlines.

5 Results

This section presents a comparative study of several partitioning scheduling algorithms with regard to the criteria introduced in Section 3.4.

The evaluation is structured as follows:

1. We study the suboptimality of FP over EDF in terms of success ratio in a multiprocessor environment.
2. In the same way, we evaluate the suboptimality of each heuristic compared with the optimal placement.
3. We determine the success ratio of each test when associated with partitioning heuristics.
4. For each given test, we determine the sorting criterion that maximizes its schedulability when associated with partitioning heuristics.
5. We compare the success ratio, number of processors used and available spare capacity of all heuristics (all tests and task set sorting criteria included).
6. Based on the best heuristic determined previously, we find the best couple heuristic-task sorting criterion.

Firstly, we have to define the concept of *suboptimality degree*. The suboptimality degree will be computed as a function of two parameters, p_1 and p_2 . It is defined as follows:

Definition 1 (Suboptimality Degree) *The degree by which the success ratio of p_1 is overpassed by the one of p_2 ($p_2 \geq p_1$ and $p_2 > 0$):*

$$sd(p_1, p_2) = \frac{\text{SuccessRatio } p_2 - \text{SuccessRatio } p_1}{\text{SuccessRatio } p_2} \cdot 100.$$

Smaller the value of $sd(p_1, p_2)$, better the performance of p_1 according to the one of p_2 .

5.1 Suboptimality of FP over EDF

The degree of suboptimality of FP schedulers according to EDF has been previously analyzed in the uniprocessor case [11]. Our study determines this degree for the multiprocessor scenario (through simulations) in relation with the total density of the task set. Figure 1 shows the simulation results as follows:

- (1) $sd(\text{OPA}[\text{FP}], \text{OPA}[\text{EDF}])$: the suboptimality degree for the case of an optimal task assignment;
- (2) $sd(\text{DM-ABRTW}, \text{EDF-BHR})$: the suboptimality degree for the case where the same exact tests are combined with all 4 heuristics.

The maximum gain of EDF over FP in the case of the optimal task assignment is 93% for high total density task sets.

When the exact tests are associated with the 4 heuristics the suboptimality degree of FP over EDF slightly increases. Though, the two curves have generally the same shape which means that the heuristics do not influence importantly the suboptimality degree of the schedulability tests.

5.2 Suboptimality of Heuristics

By definition, a heuristic is a suboptimal solution. In this section, we present the suboptimality degree of each heuristic according to the optimal task assignment. The associated schedulability test is EDF-BHR and the simulations results include all sorting criteria. Figure 2 shows the results computed as follows:

$$sd(\text{Heuristic}, \text{OPA}).$$

First of all, we should mention that, in terms of complexity, the four heuristics can be listed in decreasing order as follows: BF and WF (equal complexities), FF and finally, NF.

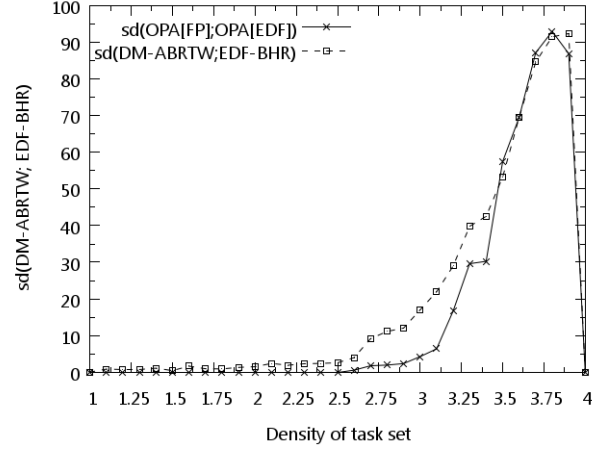


Figure 1: Heuristic - FP/EDF Suboptimality

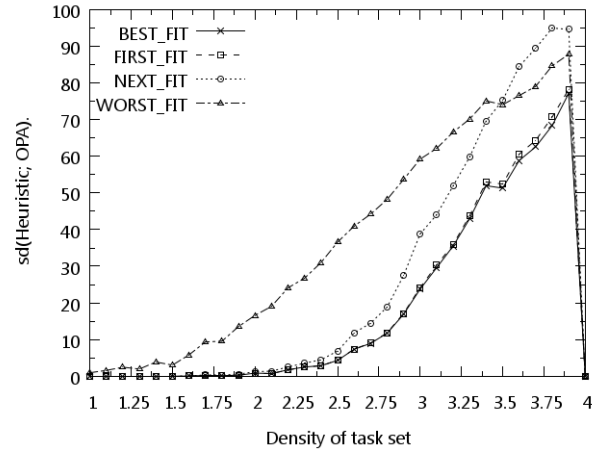


Figure 2: Suboptimality of heuristics

Figure 2 shows that for task sets with the total density bounded by half the capacity of the platform, BF, FF and NF perform the same. As NF is the least complex, it is more convenient to choose it in this case. For the scenario where the total density exceeds half of the platform capacity, BF is the best choice. Taking into account the very slight difference between FF and BF and the fact that FF has a lower complexity, FF should be also considered.

5.3 Choosing a schedulability test

In this section, we analyze the success ratio of schedulability tests for all possible combinations with the 4 heuristics and the 8 sorting criteria. The analysis is divided in three subsections: firstly, EDF tests, secondly, FP tests and finally, a comparison between the best performers of EDF and FP.

5.3.1 EDF

For *implicit deadlines*, all EDF tests reduce to EDF-LL.

For the case of *constrained deadlines* and total task set density inferior to half of the platform capacity, the two tests have the same performance as seen in Figure 3.

So, EDF-BF is the best option in this case because of its polynomial complexity. For the case with the total density exceeding half of the platform capacity, EDF-BHR is a better choice despite its pseudo-polynomial complexity.

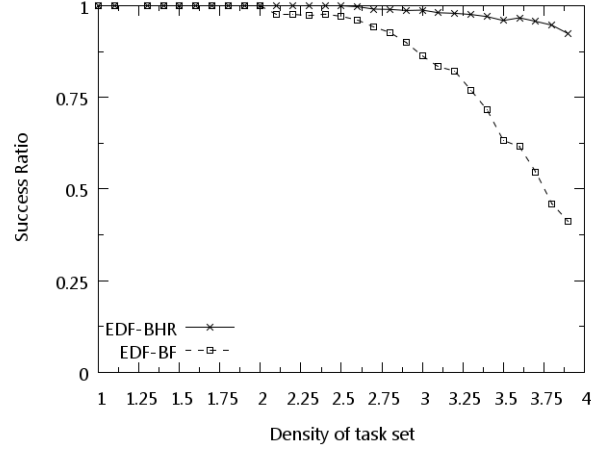


Figure 3: EDF - Constrained Deadlines

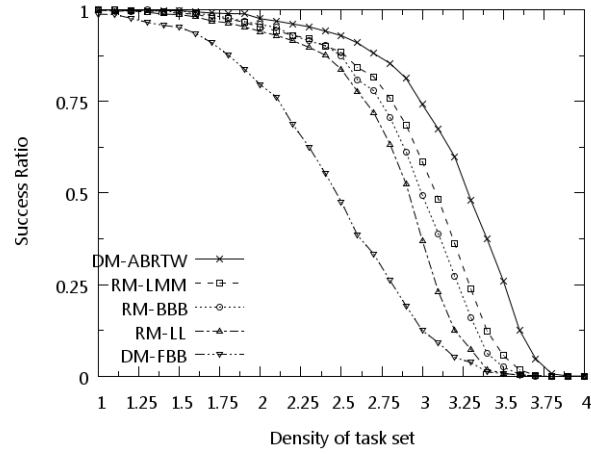


Figure 4: FP - Implicit Deadlines

5.3.2 Fixed Priority

For *implicit deadlines*, all the FP tests were taken into account during simulations. As DM-ABRTW is an exact test, it has the best performance even when associated with heuristics. For the sufficient tests, we can identify a decreasing order of their performance: RM-LMM is the best sufficient test followed by RM-BBB (which outperforms RM-LL as in the uniprocessor case); and, RM-LL performs better than DM-FBB, as is shown in Figure 4.

For the *constrained deadlines* case, DM-ABRTW is an exact schedulability test which, naturally, performs better than the only sufficient DM-FBB (Figure 5).

5.4 Choosing a Task Sorting Criterion

This section deals with the impact of a task sorting criterion on the success ratio of schedulability tests. In the corresponding graphs, *D* means *decreasing* and *I* means *increasing*. Figure 6 shows for EDF-BHR test the sorting criteria that maximize its success ratio: Decreasing Utilization and Decreasing Density.

The simulation results showed that all tests (based either on EDF or FP and excluding DM-FBB) have the same sorting criteria which maximizes their success ratio as EDF-BHR. For this reason and due to space constraints, we did not add the corresponding graphs in this paper.

DM-FBB test performs the best when associated with Period or Deadline Increasing order as seen in Figure 7 with a better performance when associated with Period Increasing order.

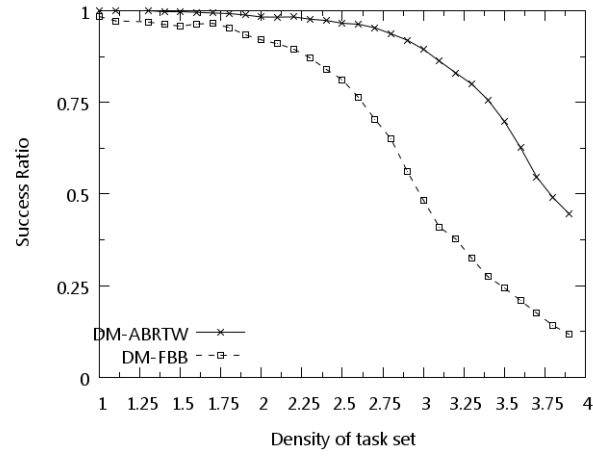


Figure 5: FP - Constrained deadlines

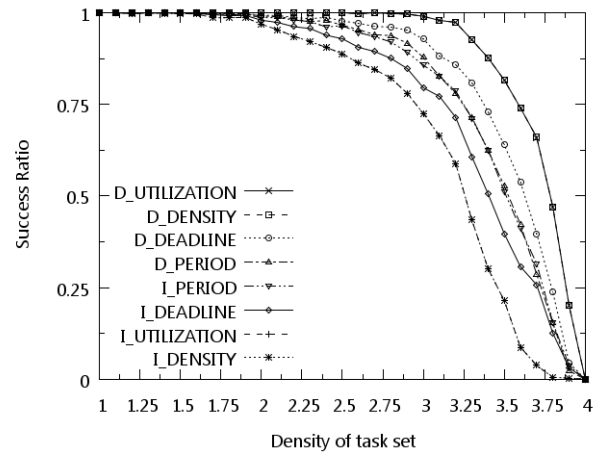


Figure 6: EDF-based Sorting Criteria

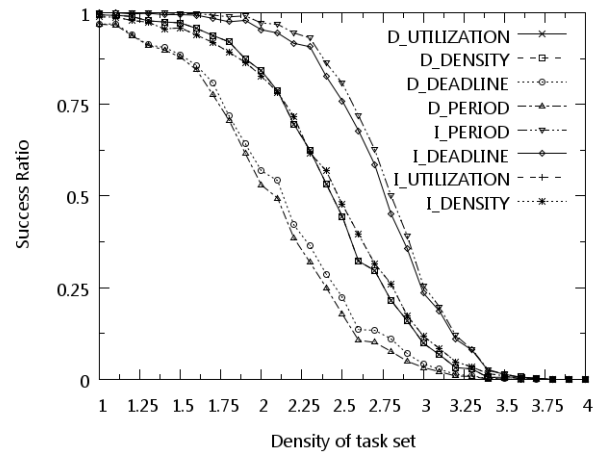


Figure 7: FBB Sorting Criteria

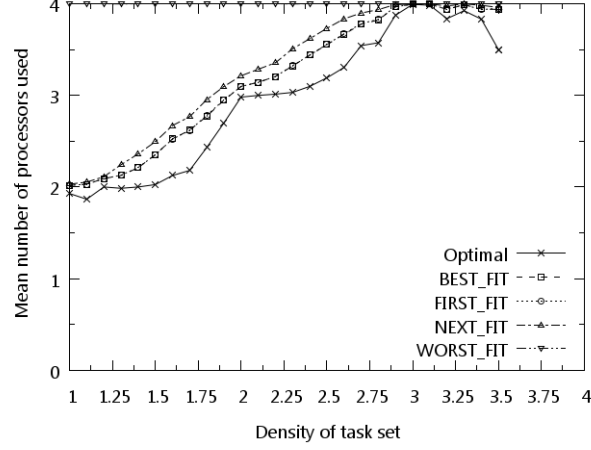


Figure 8: Minimum number of processors

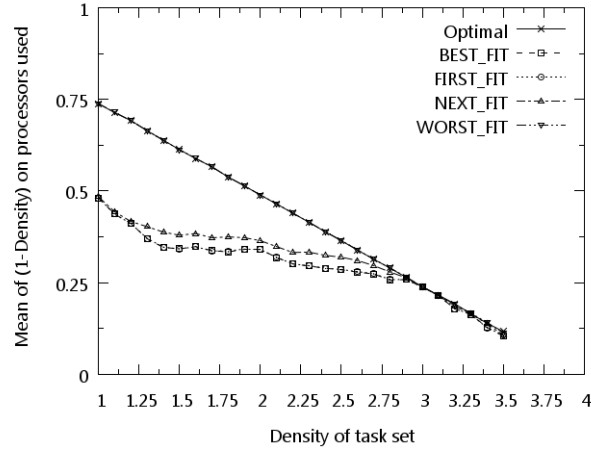


Figure 9: Unutilized capacity - $1 - \Lambda_\tau$

5.5 Choosing an Heuristic

In this section we evaluate the performance of the heuristics according to certain evaluation criteria. In this analysis each heuristic is combined with all the schedulability tests and the task sorting criteria.

Minimum Number of Used Processors. As seen in Figure 8, the heuristic that uses the smallest number of processors is FF and the one that uses the largest is WF.

Available Spare Capacity on Processors. As WF utilizes the maximum number of processors, the available spare capacity is also maximized. Figure 9 shows that WF behaves as an optimal placement according to the $1 - \Lambda_\tau$ criterion.

Also for the $1 - \text{Load}(\tau)$ criterion, WF has the closest behavior to the optimal task assignment, as shown in Figure 10.

In Section 3.1, we presented two cases for multiprocessor scheduling problem. According to the simulation results presented above, we can conclude:

- if we want to minimize the number of used processors, the best heuristic is BF which slightly outperforms FF.
- if we want to ensure an execution time slack (for the case where there is a risk to encounter software or hardware errors), the most suitable heuristic is WF with a behavior close to the one of an optimal assignment.

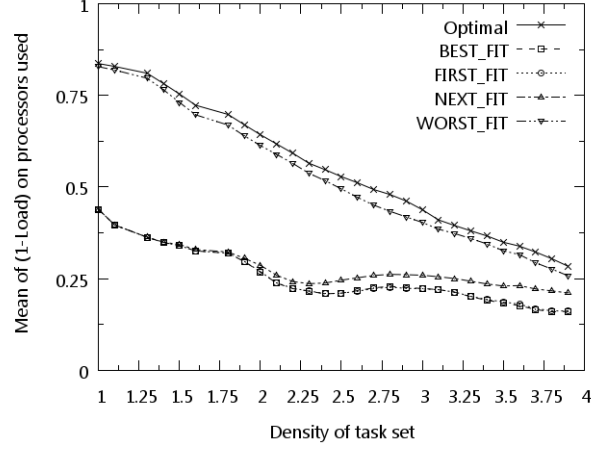


Figure 10: Unutilized capacity - $1 - \text{Load}(\tau)$

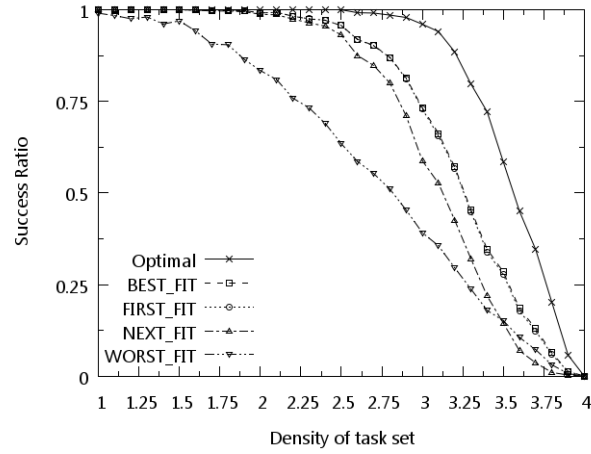


Figure 11: Heuristic - Success Ratio

Success Ratio. In Figure 11 we can observe that the success ratio of partitioning heuristics (when combined with all the schedulability tests and all the sorting criteria) follows the same performance order as in Figure 2: BF, FF, NF, WF. Taking into account the complexity of the heuristics and the density of the task set, we can choose: NF, if task set requires no more than 50% of the platform capacity for execution (due to its low complexity) or, if the task set requires more than this 50% bound, BF should be used for task assignment on processors.

5.6 Choosing a task sorting criteria for the best heuristic

According to Section 5.5, the best heuristic are BF and FF with the same success ratio. Due to its low complexity, FF is usually considered when designing partitioning algorithms. It is generally agreed that the best association partitioning heuristic — sorting criterion is FFDU (First Fit Decreasing Utilization).

Figure 12 shows that for task sets with the total density inferior to 75% of the platform capacity, all sorting criteria give the same performance. However, for task sets with total density higher than 75% of the platform capacity, Decreasing Utilization (for implicit deadlines) and Decreasing Density (for constrained deadlines) exhibit the best behavior.

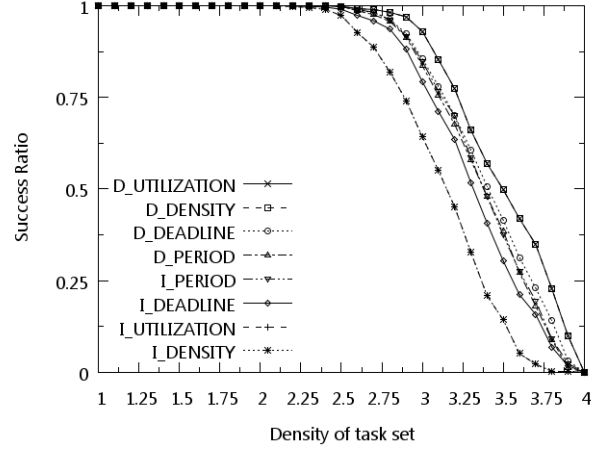


Figure 12: Task sorting criteria for FF

6 Conclusion

The main conclusion of our study is that given a task set and a processor platform, the schedulability test, the heuristic and the sorting criteria that compose the partitioning algorithm, have to be chosen according to:

- the total task density;
- the concern to:
 - minimize the number of processors;
 - ensure an execution time slack.

For the choice of schedulability tests, we tend to select exact ones due to their better performance. Furthermore, given that EDF dominates any FP scheduler, we are likely to consider EDF. However, when choosing a schedulability test, it is important to observe the task set total density, as, for values bounded by half of the platform capacity, exact tests perform the same as sufficient ones which is an advantage given the lower implementation complexity.

The task set total density is also important when choosing an heuristic: in the case of minimizing the number of processors, for task sets with density bounded by half of the platform capacity, NF is the best choice, but for task sets with total density higher than 50% of the platform capacity, BF has to be considered. Furthermore, we want to ensure an execution slack, WF has a behavior close to the one of optimal task assignment.

From the simulations, we have deduced that the best task sorting criteria is Decreasing Density for all schedulability tests, even if the priority assignment is based on the deadline (EDF, DM) or the period (RM). There is one exception, DM-FBB which has the best performance when associated to Increasing Period sorting criteria.

According to the assertions above, one can guide its choices regarding the test, the heuristic and the task sorting criteria for scheduling a real-time task set on a multiprocessor platform.

7 Future Work

In the future, we firstly want to extend this study to heterogeneous platforms. Secondly, using the same approach as for this analysis, we want to evaluate the performance of a new class of multiprocessor algorithms called *semi-partitioning*. As this technique combines the 2 previously existing ones (global and partitioning), it would be interesting to widen the analysis with new evaluation criteria such as the number of context switches (migrations and preemptions).

References

- [1] B. Andersson, S. Baruah, and J. Jonsson. Static-priority scheduling on multiprocessors. In *22nd IEEE Real-Time Systems Symposium*, pages 193–202, 2001.
- [2] N. Audsley, A. Burns, M. Richardson, K. Tindell, and J. Wellings. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Engineering Journal*, 8:284–292, 1993.
- [3] N. C. Audsley, A. Burns, M. F. Richardson, and A. J. Wellings. Real-time scheduling: the deadline-monotonic approach. In *IEEE Workshop on Real-Time Operating Systems and Software*, pages 133–137, 1991.
- [4] T. P. Baker. Comparison of empirical success rates of global vs. partitioned fixed-priority and EDF scheduling for hard real time. Technical Report TR-050601, The Florida State University, 2005.
- [5] T. P. Baker. A comparison of global and partitioned EDF schedulability tests for multiprocessors. In *International Conf. on Real-Time and Network Systems*, pages 119–127, 2006.
- [6] T. P. Baker and S. Baruah. *Handbook of Real-Time and Embedded Systems*, chapter Schedulability Analysis of Multiprocessor Sporadic Task Systems, pages 3.1–3.15. Chapman, 2008.
- [7] S. Baruah. Techniques for multiprocessor global schedulability analysis. In *28th IEEE International Real-Time Systems Symposium*, pages 119–128, 2007.
- [8] S. Baruah and N. Fisher. The partitioned multiprocessor scheduling of deadline-constrained sporadic task systems. *IEEE Trans. on Computers*, 55(7):918–923, 2006.
- [9] S. Baruah, R. Howell, and L. Rosier. Algorithms and complexity concerning the preemptive scheduling of periodic real-time tasks on one processor. *Real-Time Systems*, 2:301–324, 1990.
- [10] E. Bini, G. C. Buttazzo, and G. M. Buttazzo. Rate monotonic analysis: The hyperbolic bound. *IEEE Transactions on Computers*, 52(7):933–942, 2003.
- [11] R. Davis and A. Burns. Exact quantification of the sub-optimality of uniprocessor fixed-priority pre-emptive scheduling. *Real-Time Systems*, (3):211–258, 2009.
- [12] R. Devillers and J. Goossens. Liu and Layland’s schedulability test revisited. *Information Processing Letters*, 73(5–6):157–161, 2000.
- [13] N. Fisher, S. Baruah, and T. Baker. The partitioned scheduling of sporadic tasks according to static-priorities. In *Proceedings of the 18th Euromicro Conference on Real-Time Systems*, pages 118–127, Dresden, Germany, 2006.
- [14] N. Fisher and S. K. Baruah. A fully polynomial-time approximation scheme for feasibility analysis in static-priority systems with arbitrary relative deadlines. In *Proceedings of the 17th Euromicro Conference on Real-Time Systems*, pages 117–126, 2005.
- [15] L. George and J. Hermant. A norm approach for Partitioned EDF Scheduling of Sporadic Task Systems. *Proceedings of the 21st Euromicro Conference on Real-Time Systems*, Dublin, Ireland, July 2009.
- [16] S. Lauzac, R. G. Melhem, and D. Mossé. An efficient RMS admission control and its application to multiprocessor scheduling. In *12th International Parallel Processing Symposium*, pages 511–518, 1998.
- [17] L. C. Liu and W. Layland. Scheduling algorithms for multi-programming in a hard real time environment. *Journal of ACM*, 20(1):46–61, January 1973.
- [18] V. Zhirnov, R. Cavin, J. Hutchby, and G. Bourianoff. Limits to binary logic switch scaling—a Gedanken model. In *IEEE*, volume 9, pages 1934–1939, 2003.